# UTPlaceF 3.0: A Parallelization Framework for Modern FPGA Global Placement

**Wuxi Li**, Meng Li, Jiajun Wang, and David Z. Pan

University of Texas at Austin

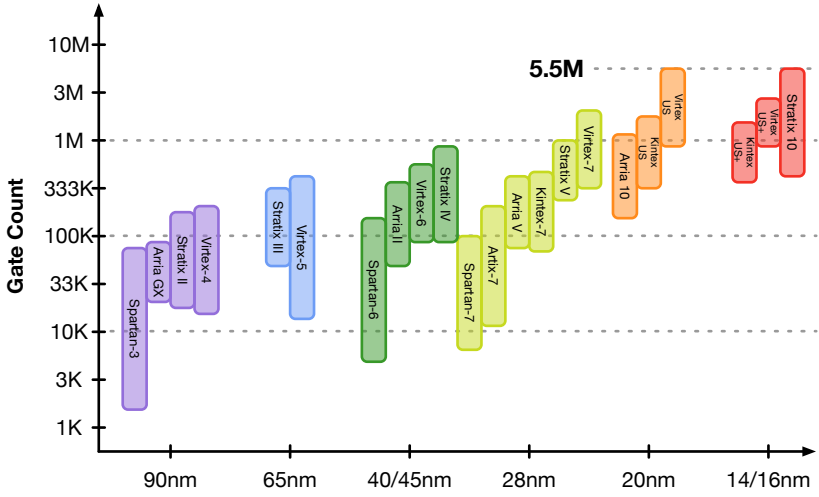
*wuxili@utexas.edu*

November 14, 2017

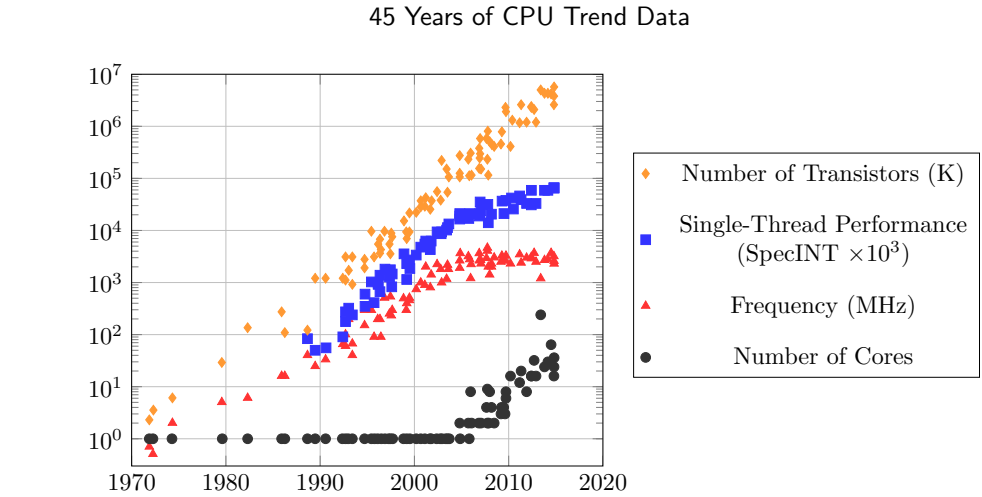

# Outline

# Introduction: FPGA Scaling

15 Years of Xilinx/Intel(Altera) FPGA Size Trend



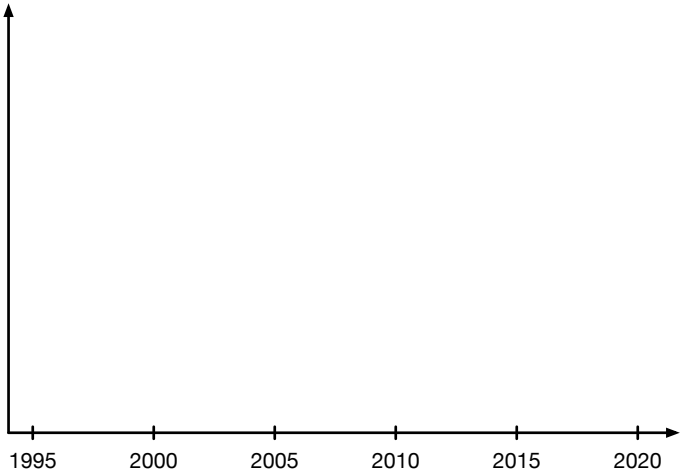
# Introduction: The Era of Multi-Core CPUs




Source: Original data up to the year 2010 was collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. Data for 2010-2015 was collected by K. Rupp.

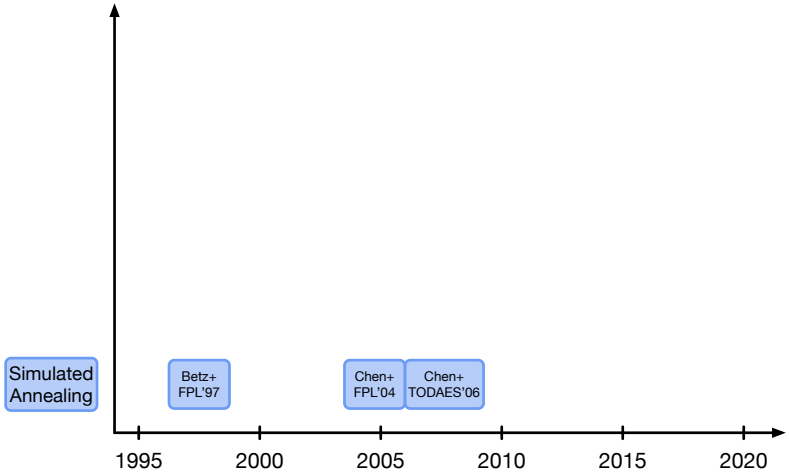
# The Evolution of FPGA Placement

20 Years of Selected Academic Papers on FPGA Placement



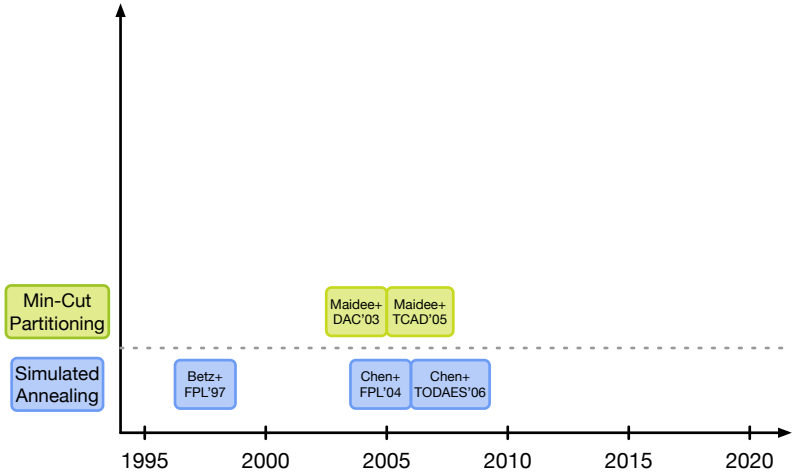
# The Evolution of FPGA Placement

20 Years of Selected Academic Papers on FPGA Placement



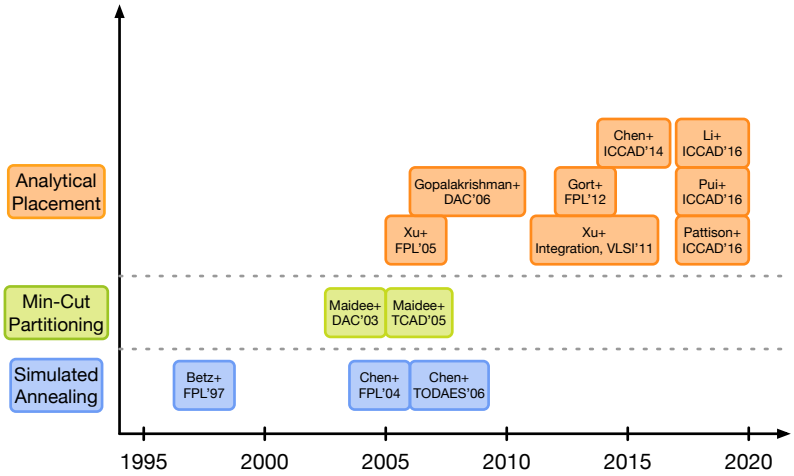
# The Evolution of FPGA Placement

20 Years of Selected Academic Papers on FPGA Placement



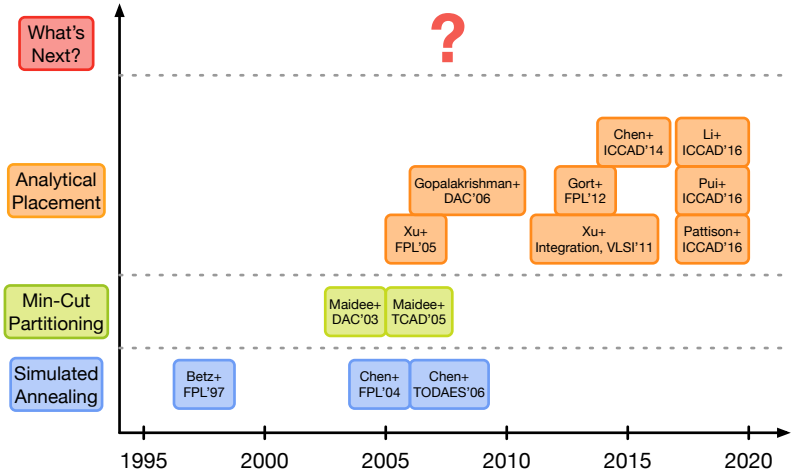
# The Evolution of FPGA Placement

20 Years of Selected Academic Papers on FPGA Placement



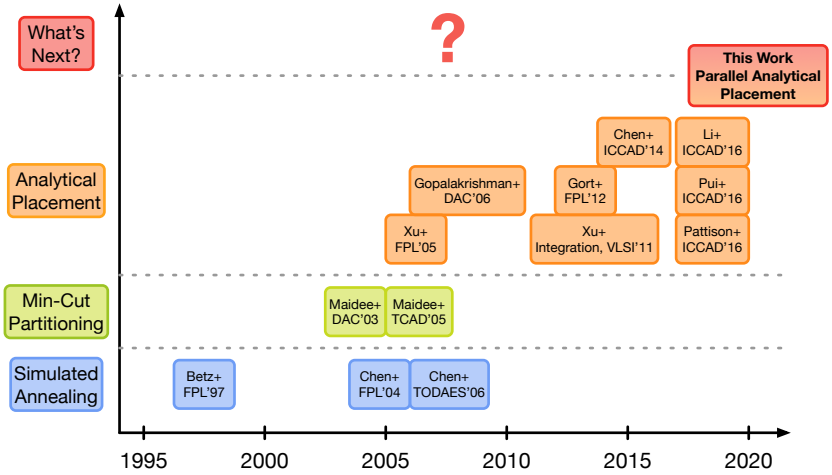
# The Evolution of FPGA Placement

20 Years of Selected Academic Papers on FPGA Placement



# The Evolution of FPGA Placement

20 Years of Selected Academic Papers on FPGA Placement



# A Representative Quadratic Placement Flow

Quadratic placers minimize cost function

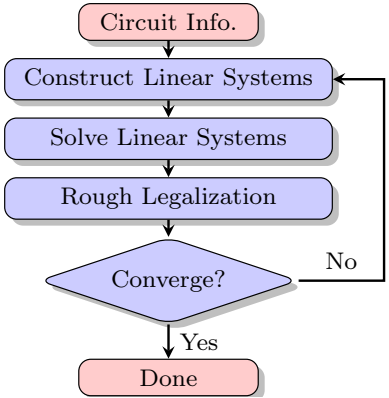
- $W(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{x}^T Q_x \mathbf{x} + \mathbf{c}_x^T \mathbf{x} + \frac{1}{2}\mathbf{y}^T Q_y \mathbf{y} + \mathbf{c}_y^T \mathbf{y} + const$

It is equivalent to solving

- $Q_x \mathbf{x} = -\mathbf{c}_x$
- $Q_y \mathbf{y} = -\mathbf{c}_y$

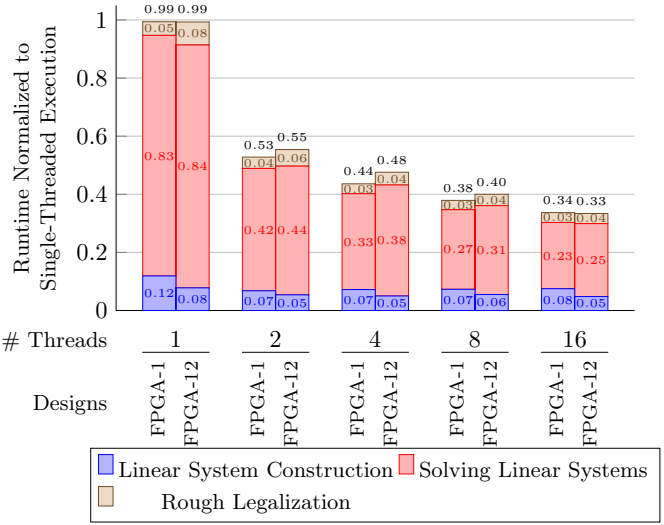
(Conjugate Gradient (CG) method)

Rough legalization [Kim+, TCAD'12] to remove cell overlapping



# A Simple Parallelization Attempt

- 100K-cell (FPGA-1)
  1.1M-cell (FPGA-12)
- Intel Math Kernel Library (MKL) for CG
- Solve X and Y in parallel, if #thread $\geq 2$
- Enable parallelization for matrix-vector operations in CG, if #thread $> 2$
- Use parallel sorting algorithms in libstdc++ parallel mode for rough legalization

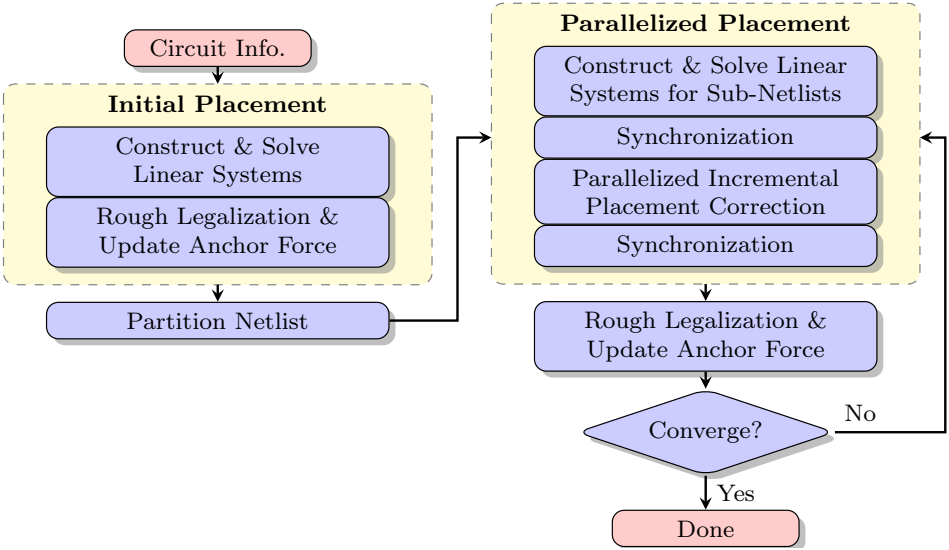


# Previous Works

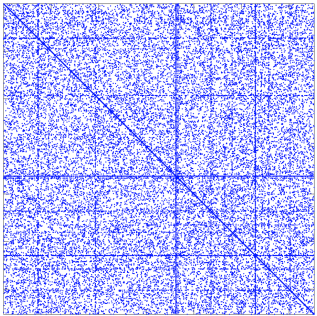
**POLAR 3.0 [Lin+, ICCAD'15]**

- ASIC quadratic placer
- Solve X and Y directions in parallel
- Solve geometric partitions in parallel
- Periodically solve flat placement
- 4X speedup with 16 threads
- Small to medium quality degradation

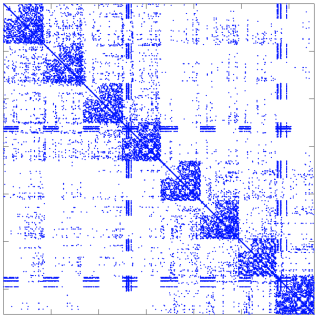
# UTPlaceF 3.0 Overview



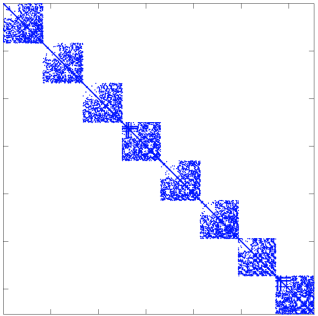
# Block-Jacobi Preconditioning



The matrix $Q$



$Q$ after 8-way partitioning by row/column permutation



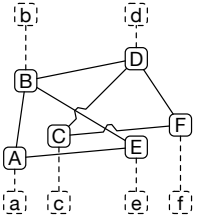
The block-Jacobi preconditioner $\widehat{Q}$ of $Q$

$$Q\mathbf{x} = -\mathbf{c} \quad \Longrightarrow \quad \widehat{Q}_i\mathbf{x}_i = -\mathbf{c}_i, \forall i = 1 \ldots 8$$

# Physical Interpretation of Block-Jacobi Preconditioning

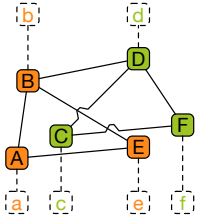
$$Q \quad \mathbf{x} \quad -\mathbf{c}$$

| | A | B | C | D | E | F | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | 3 | 1 | | | 1 | | $x_A$ | = | a |
| B | 1 | 4 | | | 1 | | $x_B$ | | b |
| C | | | 3 | 1 | | 1 | $x_C$ | | c |
| D | | | 1 | 4 | | 1 | $x_D$ | | d |
| E | 1 | 1 | | | 3 | | $x_E$ | | e |
| F | | | 1 | 1 | | 3 | $x_F$ | | f |



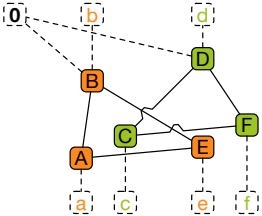

$$Q \quad \mathbf{x} \quad -\mathbf{c}$$

| | A | B | E | C | D | F | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | 3 | 1 | 1 | | | | $x_A$ | = | a |
| B | 1 | 4 | 1 | | 1 | | $x_B$ | | b |
| E | 1 | 1 | 3 | | | | $x_E$ | | e |
| C | | | | 3 | 1 | 1 | $x_C$ | | c |
| D | | 1 | | 1 | 4 | 1 | $x_D$ | | d |
| F | | | | 1 | 1 | 3 | $x_F$ | | f |




$$\widehat{Q} \quad \mathbf{x} \quad -\mathbf{c}$$

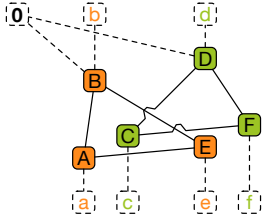
| | A | B | E | C | D | F | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | 3 | 1 | 1 | | | | $x_A$ | = | a |
| B | 1 | 4 | 1 | | | | $x_B$ | | b |
| E | 1 | 1 | 3 | | | | $x_E$ | | e |
| C | | | | 3 | 1 | 1 | $x_C$ | | c |
| D | | | | 1 | 4 | 1 | $x_D$ | | d |
| F | | | | 1 | 1 | 3 | $x_F$ | | f |



# Placement-Driven Block-Jacobi Preconditioning (PDBJP)

$$\widehat{Q} \quad \mathbf{x} \quad -\mathbf{c}$$

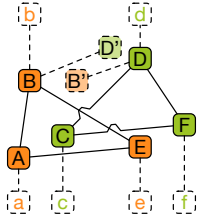
| | A | B | E | C | D | F | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | 3 | 1 | 1 | | | | $x_A$ | = | a |
| B | 1 | 4 | 1 | | | | $x_B$ | | b |
| E | 1 | 1 | 3 | | | | $x_E$ | | e |
| C | | | | 3 | 1 | 1 | $x_C$ | | c |
| D | | | | 1 | 4 | 1 | $x_D$ | | d |
| F | | | | 1 | 1 | 3 | $x_F$ | | f |




Block-Jacobi Preconditioning

$$\widehat{Q} \quad \mathbf{x} \quad -\widehat{\mathbf{c}}$$

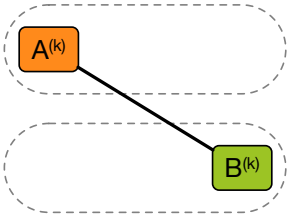
| | A | B | E | C | D | F | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | 3 | 1 | 1 | | | | $x_A$ | = | a |
| B | 1 | 4 | 1 | | | | $x_B$ | | b + $x_D$' |
| E | 1 | 1 | 3 | | | | $x_E$ | | e |
| C | | | | 3 | 1 | 1 | $x_C$ | | c |
| D | | | | 1 | 4 | 1 | $x_D$ | | d + $x_B$' |
| F | | | | 1 | 1 | 3 | $x_F$ | | f |



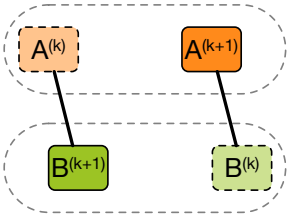

Placement-Driven
Block-Jacobi Preconditioning

# Can We Do Better?

- PDBJP is effective for a small number of partitions.
- Its quality degrades dramatically as the partition count increases.
- The quality degradation in PDBJP is introduced by the discrepancy of inter-partition nets in different partitions.



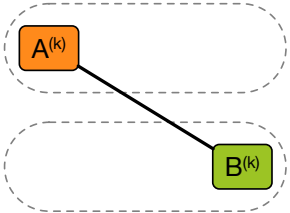

An inter-partition net in an intermediate placement



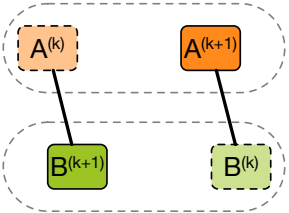

The solution after applying PDBJP with large discrepancy

# Parallelized Incremental Placement Correction (PIPC)

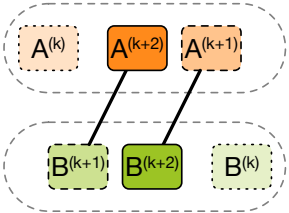
- A local and smooth perturbation of the post PDBJP placement
- Reduce the discrepancy of inter-partition nets




An inter-partition net in an intermediate placement




The solution after applying PDBJP with large discrepancy




An incremental improvement that reduces the discrepancy

# Mathematical Formulation of PIPC

PIPC is based on the idea of Additive Correction Multigrid Method
The original problem we are trying to solve is

$$Q\mathbf{x} = -\mathbf{c}. \tag{1}$$

Let $\mathbf{x}^{(0)}$ be the solution of the PDBJP linear system (1), that is

$$\widehat{Q}\mathbf{x}^{(0)} = -\widehat{\mathbf{c}}. \tag{2}$$

If we can find $\Delta\mathbf{x}$ satisfying

$$Q\Delta\mathbf{x} = \mathbf{r}^{(0)} = -\mathbf{c} - Q\mathbf{x}^{(0)},$$

the solution of the linear system (1) will be $\mathbf{x}^{(0)} + \Delta\mathbf{x}$.

# Mathematical Formulation of PIPC

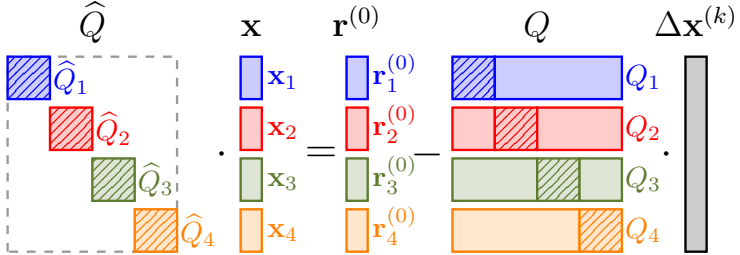
$\Delta x$ can be approached by the iterative method

$$\Delta \mathbf{x}^{(k+1)} = \Delta \mathbf{x}^{(k)} + \widehat{Q}^{-1}(\mathbf{r}^{(0)} - Q\Delta \mathbf{x}^{(k)})$$

Computing $\widehat{Q}^{-1}(\mathbf{r}^{(0)} - Q\Delta \mathbf{x}^{(k)})$ is equivalent to solving

$$\widehat{Q}\mathbf{x} = \mathbf{r}^{(0)} - Q\Delta \mathbf{x}^{(k)},$$

which can be parallelized as well.



# Runtime and Quality Tradeoff of PIPC

**Theorem: The Convergence of PIPC**

By picking any placement-driven block-Jacobi preconditioner of $Q$ as $\widehat{Q}$, PIPC defined by iterative method

$$\Delta \mathbf{x}^{(k+1)} = \Delta \mathbf{x}^{(k)} + \widehat{Q}^{-1}(\mathbf{r}^{(0)} - Q\Delta \mathbf{x}^{(k)})$$

is monotonically convergent for any choice of $\Delta \mathbf{x}^{(0)}$.

The Theorem reveals two important properties of PIPC:

- With sufficient iterations, the exact solution, $\mathbf{x}^{(0)} + \Delta \mathbf{x}$, can be reached
- More iterations guarantees better solutions for $Q\mathbf{x} = -\mathbf{c}$

**The runtime and quality can be perfectly traded to each other in PIPC.**

# Experiments Setup

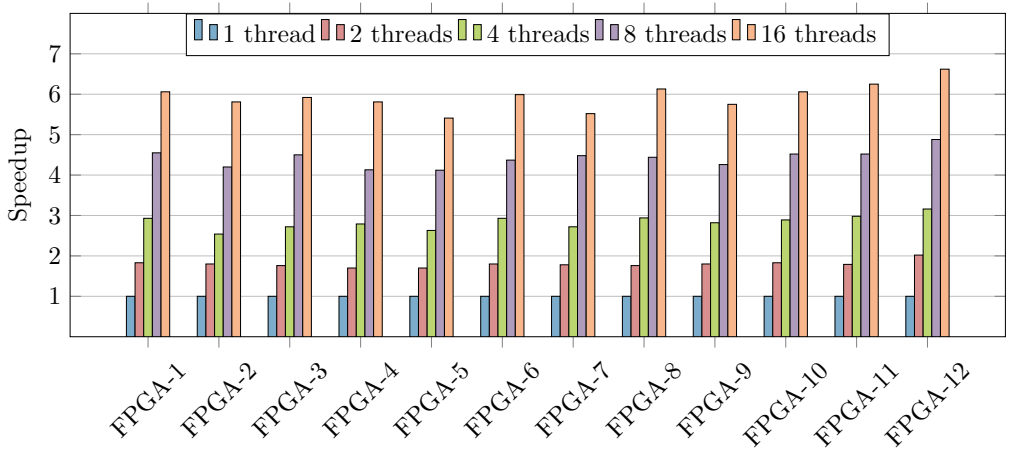
- C++
- Intel Xeon E5-2690 CPUs (2.6 GHz, 24 cores, and 30M L3 cache)
- 64 GB RAM
- Eigen 3 for linear system solving
- OpenMP 4.0 for multi-threading
- libstdc++ parallel mode for parallel sorting
- ISPD'16 FPGA placement contest benchmark suite

# Parallelization Configuration

- Solve X and Y in parallel
- Enable PDBJP for #thread $> 2$
- PIPC is only enable for $> 4$ threads
- PIPC is not called for every placement iteration for 8 threads
- Less CG iterations for PIPC compared with PDBJP

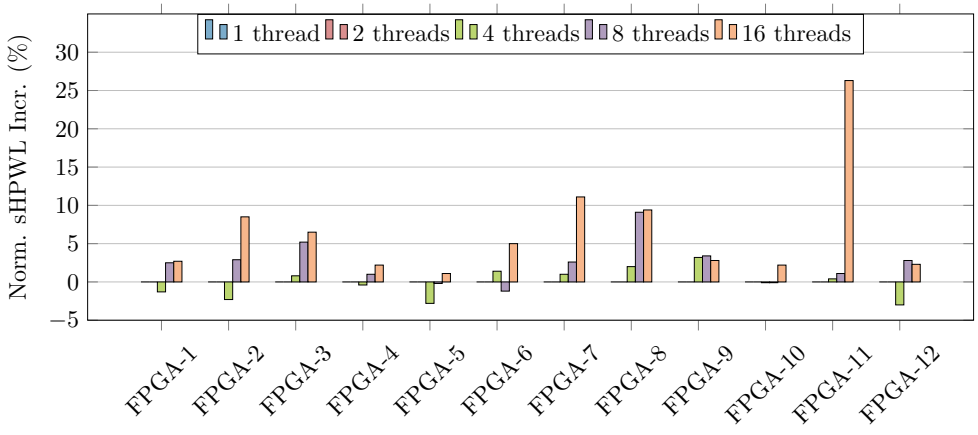
| #Threads | 1 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|---|
| # Partitions | 1 | 1 | 2 | 4 | 8 |
| # CG Iter. for PDBJP | 250 | 250 | 250 | 250 | 250 |
| # PIPC Period | - | - | - | 3 | 1 |
| # Correction Iter. of each PIPC | - | - | - | 1 | 1 |
| # CG Iter. for PIPC | - | - | - | 150 | 50 |

# Runtime Result w/o PIPC



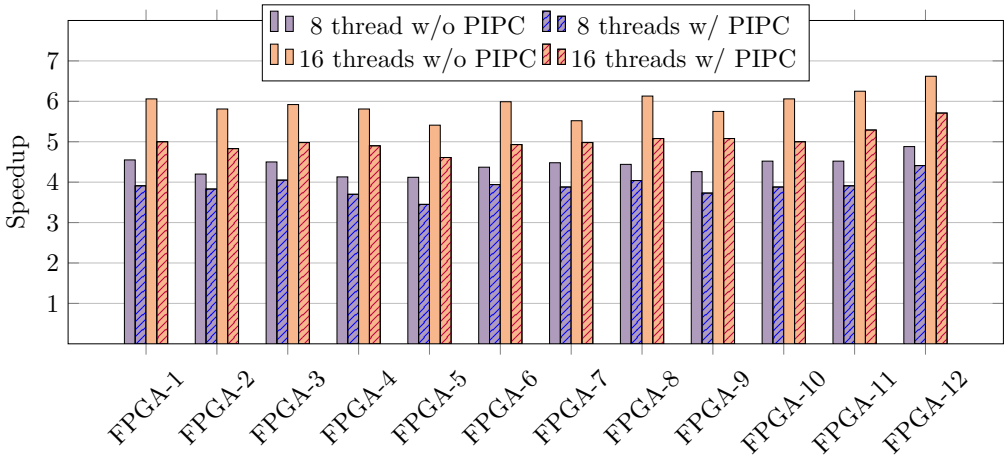

| #Threads | 1 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|---|
| Avg. Speedup | 1.00 | 1.80 | 2.83 | 4.41 | 5.92 |

# Wirelength Result w/o PIPC



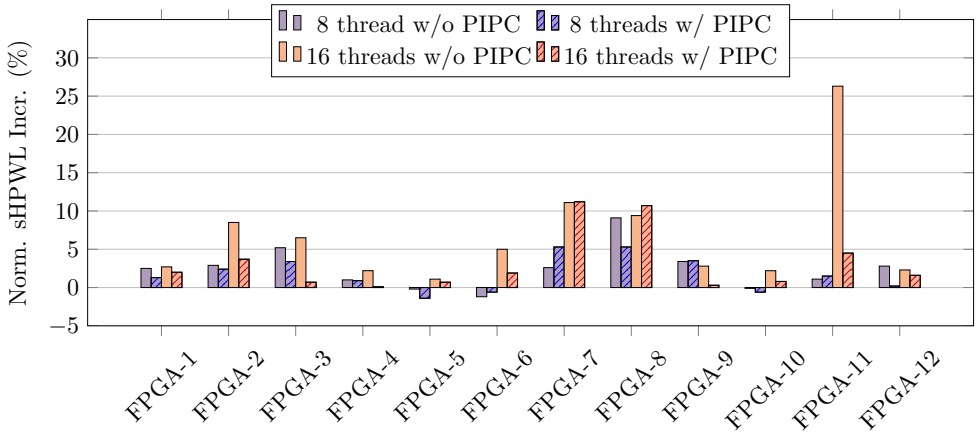

| #Threads | 1 | 2 | 4 | 8 | 16 |
|---|---|---|---|---|---|
| Avg. sHPWL Incr. | 0.0% | 0.0% | -0.1% | 2.4% | 6.5% |

# Runtime Result w/ PIPC



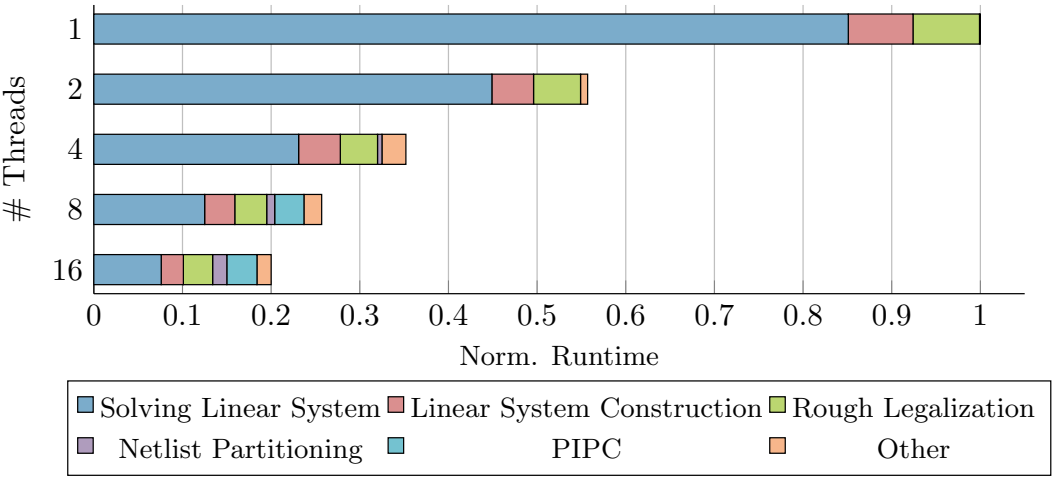

| #Threads | 8 w/o PIPC | 8 w/ PIPC | 16 w/o PIPC | 16 w/ PIPC |
|---|---|---|---|---|
| Avg. Speedup | 4.41 | 3.89 | 5.92 | 5.03 |

# Wirelength Result w/ PIPC




| #Threads | 8 w/o PIPC | 8 w/ PIPC | 16 w/o PIPC | 16 w/ PIPC |
|---|---|---|---|---|
| Avg. sHPWL Incr. | 2.4% | 1.7% | 6.5% | 3.0% |

# Runtime Breakdown



# Conclusion & Future Work

## Conclusion

- UTPlaceF 3.0, a parallelization framework for FPGA quadratic placement
- Placement-driven block-Jacobi preconditioning
- Parallelized incremental placement correction
- 5X speedup with 3.0% wirelength increase

## Futher Work

- Parallelize packing and detailed placement
- Hardware acceleration: FPGA, GPU

# Thank You!